This tutorial walks through how to become able to code the USRP using Python. I prefer Python over MATLAB, so having this option is highly beneficial. I was able to set up transmitting and receiving operations using the USRP, but unable to implement threading for transmitting and receiving.

Step 1. Go to this link: https://files.ettus.com/binaries/uhd/latest_release/ and download the needed executable for installing the most recent release of uhd. The .exe will be in the folder for Windows10 or Windows11 depending on the operating system

Step 2. Ensure you have Python 3.10 or 3.10.2 installed and it is on the PATH

Step 3. Run this line in the command prompt:
"C:\Path\To\Python\AppData\Local\Programs\Python\Python310\python.exe" -m pip install uhd==4.9.0.0 or in VS Code: python -m pip install uhd==4.9.0.0
I have not tested VSCode, I did this entire tutorial in the command prompt.
The reason the Python .exe has to come before the pip command is because windows defaults to the Microsoft Store version of Python 3, since we are trying to ensure this works in VS Code we must use the downloaded version of Python meant for VS Code.

Step 4. Since the USRP x410 does not have an IP address assigned by default, the following DHCP server had to be downloaded and configured to automatically assign the IP address to the USRP: dhcpserver.de

Step 5. Configuring the DHCP Server

- Search Network Connections on Windows
- Right click on ethernet and select properties.
- Double click on TCP/IPv4, set your address to be 192.168.10.1 and Subnet to be 255.255.255.0
- Configure the DHCP Server and use a range of 192.168.10.2 - 192.168.10.100, finish setup of the DHCP server and run it
- Open the dhcpsrv.exe to make sure the server is running and open the command prompt on windows and type in “ping 192.168.10.2” if the ping succeeds the USRP is working.

Step 6. Verify the USRP device shows up using either of the commands provided: "C:\Program Files\UHD\bin\uhd_find_devices.exe" or "C:\Program Files\UHD\bin\uhd_usrp_probe.exe"

Step 7. Use the receive test script provided to test if the USRP is able to be programmed in Python. If the device prints a line out to the terminal claiming the device claimed=True there are addition steps to resolve this issue

Resolving Claiming Issue:

Step 1. Run "C:\Program Files\UHD\bin\uhd_find_devices.exe”

Step 2. Run "C:\Program Files\UHD\bin\uhd_usrp_probe.exe" --args "addr=192.168.10.2" or "C:\Program Files\UHD\bin\uhd_usrp_probe.exe" --args "addr=192.168.10.2,force" You may need to reopen command prompt with it running as an administrator to get these commands to work

Step 3. The USRP should have claimed = False, allowing you to interact with it using VS Code and not the command prompt

Future Work: Implement Threading to allow transmitting and receiving using the USRP. Was unable to pick up the Pluto's transmission while receiving with USRP. Was able to pick up the USRP transmitting while receiving with Pluto.

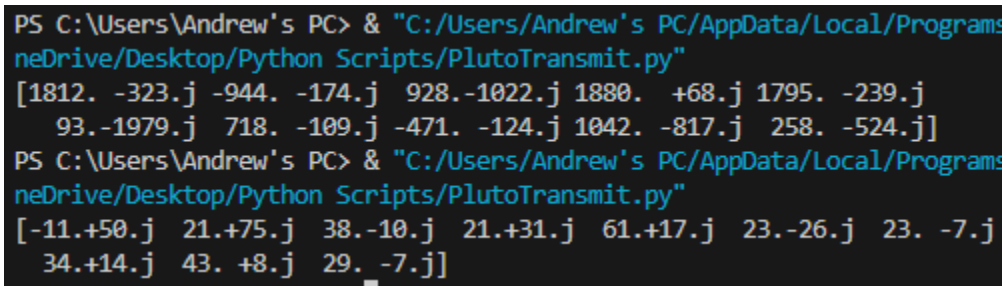

Figure 1. Pluto Receiving USRP Transmission and Pluto Receiving Noise

The first receive array shows huge values of 1812 and -1022j, this is when the USRP is transmitting. The second receive is after transmission has ended and Pluto is just picking up noise. This test shows both devices work using Python, which is very exciting. The only problem is the USRP does not seem to detect Pluto's transmission. The Pluto receive code is attached below. The USRP code should be found in the github.

```
import numpy as np
import adi

sample_rate = 1e6 # Hz
center_freq = 915e6 # Hz
num_samps = 10000 # number of samples returned per call to rx()

sdr = adi.Pluto('ip:192.168.2.1')
sdr.gain_control_mode_chan0 = 'manual'
sdr.rx_hardwaregain_chan0 = 70.0 # dB
sdr.rx_lo = int(center_freq)
sdr.sample_rate = int(sample_rate)
sdr.rx_rf_bandwidth = int(sample_rate) # filter width, just set it to the
same as sample rate for now
sdr.rx_buffer_size = num_samps
```

```
samples = sdr.rx() # receive samples off Pluto
print(samples[0:10])
```